

EzTemplate Tutorial

Introduction

EzTemplate is designed to help users easily create templates they use to generate plots in VCS.

EzTemplate's first sub-package, Multi, is designed to create templates for multiple plots per page.

EzTemplate is easy to use in a two step process:

1. Initialize the object and set up values to use.
2. Get the VCS template object.

All EzTemplate objects have a preview method that generates a .gif file illustrating positions of plots.

In the following examples we only use the preview method, but we show (commented out) an example line on how to use the template in VCS. In the examples, "s" is the variable to be plotted and "iso" is the graphics method that would be used to display "s".

The Multi Object

Definitions

Multi object allows the user to display multiple plots on a single page.

In Version 1 a few assumption are made:

- The legend is either at the bottom or on the right side of the plot (always centered).

The VCS Canvas has multiple sections or attributes with default values indicated between parentheses

- Rows (3) and columns (2)
- The **margins**: **top** (0.05), **bottom** (0.075), **left** (0.033) and **right** (0.05)
 - ◆ Parameter values indicate percentage (%) of the page.
 - ◆ The bottom and right margins are used as space to display the global horizontal/vertical legend.
- The **spacing**: **horizontal** (0.05) and **vertical** (0.035)
 - ◆ Spacing defines the space between two adjacent plots on a row or a column.
 - ◆ Parameter values indicate percentage (%) of the page.
- The **legend**: **direction** ("horizontal"), **thickness** (0.2), **stretch** (0.8) and **fat** (0.05)
 - ◆ Parameter values for thickness and stretch indicate percentage (%) of the space allocated for the legend bar.
 - ◆ Direction of the legend bar can be "horizontal" or "vertical".
 - ◆ Fat: only used for local (inside the grid) plots, the parameter value indicating percentage (%) of the grid space to use for the legend bar.
- The **get** function

- ◆ Creates and returns the template object associated with the arguments **row** and **column**.
- ◆ If no row/column arguments are given, then **get** creates plots in order of each call, the plots being drawn starting from top left, going right then down.
- ◆ **legend** keyword can be passed as "local", indicating that this specific plot needs to have a legend associated with it. This reduces the area allocated for the data plotting itself, and the direction of the legend will be that of the legend.direction attribute at the time the function is called.
- The **preview** function
 - ◆ Generates the templates (if none already generated with a call to get) and outputs a preview result to the file specified by the **out** keyword.

Examples

In this example we simply plot three columns and four rows:

```
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)

for i in range(12):
    t=M.get()
    ##     x.plot(s[i],t,iso)
M.preview('test')
raw_input('Done')
```



In this example show how to plot a legend on the side of each individual plot:

```
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.legend.direction='vertical'
for i in range(12):
    t=M.get(legend='local')
```

```
##      x.plot(s[i],t,iso)
M.preview('test2')
```

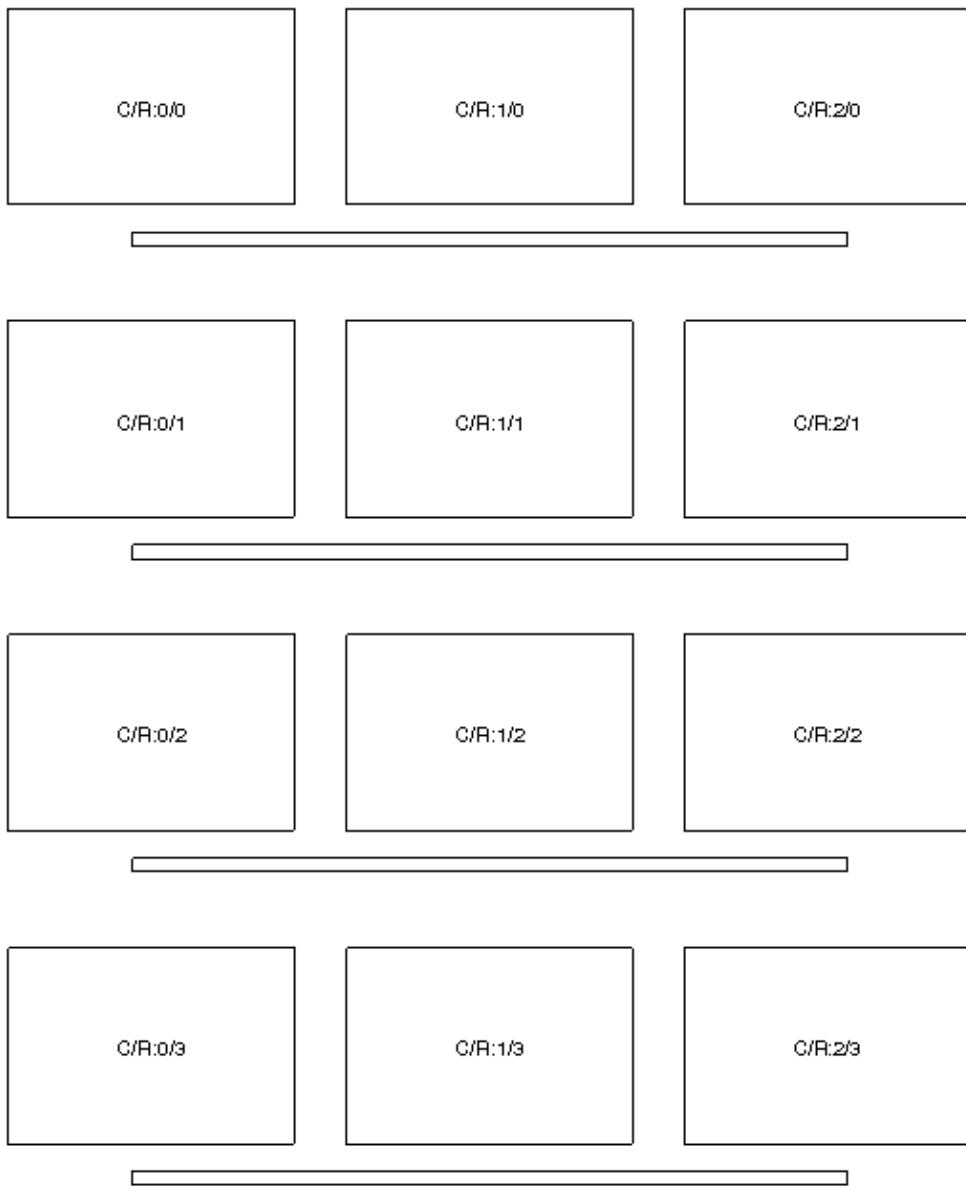


In this example we show how to draw a horizontal legend bar for each row:

```
import EzTemplate,vcs
## 12 plot one legend per row

## Initialize VCS
x=vcs.init()
```

```
M=EzTemplate.Multi(rows=4,columns=3)
M.legend.stretch=2.5 # 250% of width (for middle one)
for i in range(12):
    t=M.get(legend='local')
    if i%3 !=1:
        t.legend.priority=0 # Turn off legend
##    x.plot(s[i],t,iso)
M.preview('test3')
raw_input('Done')
```



In this example we show how to draw one legend per row, vertically:

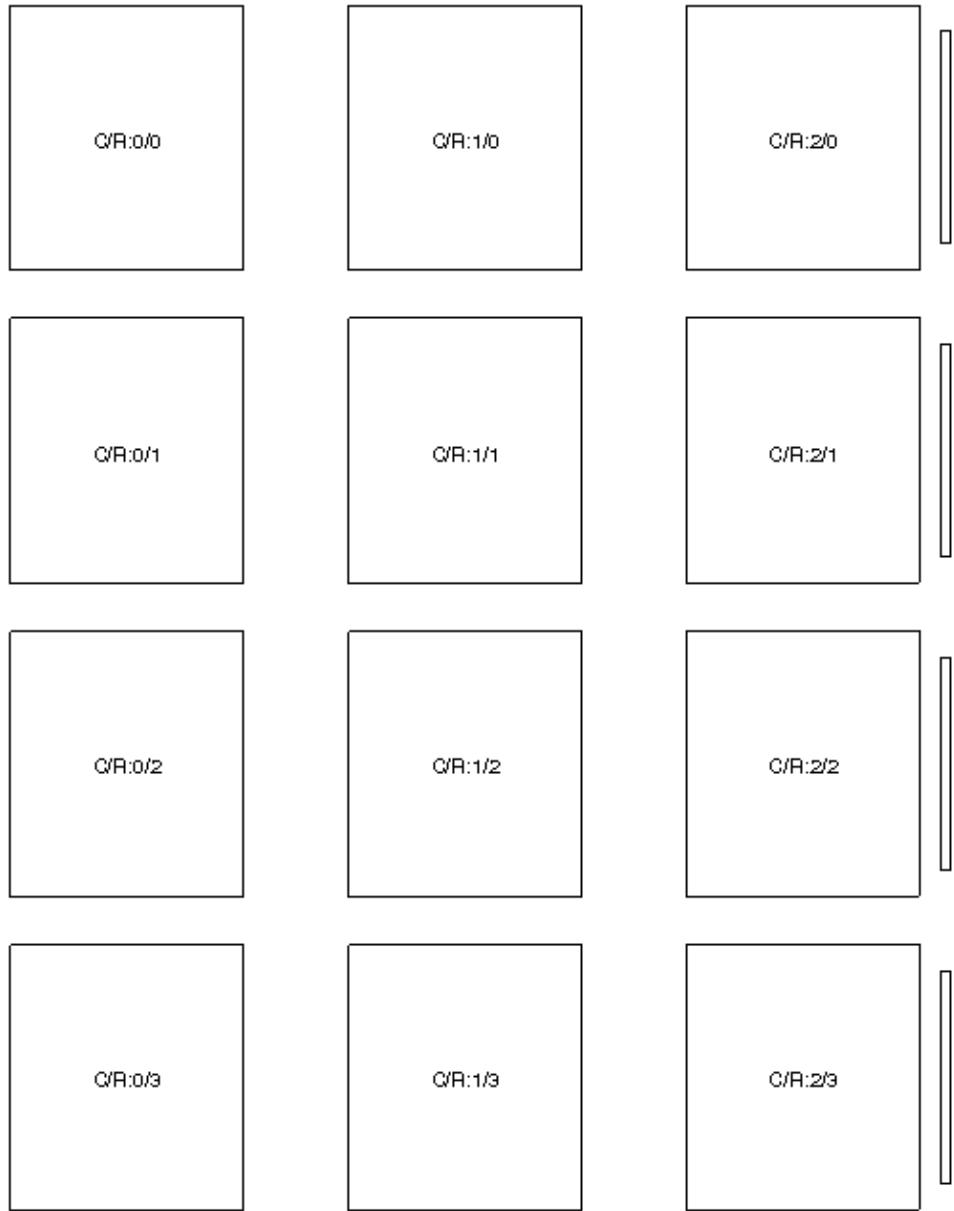
```
import cdms,EzTemplate,vcs,sys
## 12 plots 1 legend per row on the right
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.legend.direction='vertical'
for i in range(12):
```

```

t=M.get(legend='local')
if i%3 !=2:
    t.legend.priority=0 # Turn off legend
##    x.plot(s[i],t,iso)
M.preview('test3b')
raw_input('Done')

```



In this example we show how to control margins and legend thickness:

```

import cdms,EzTemplate,vcs,sys
## 12 plot playing with margins and legend thickness

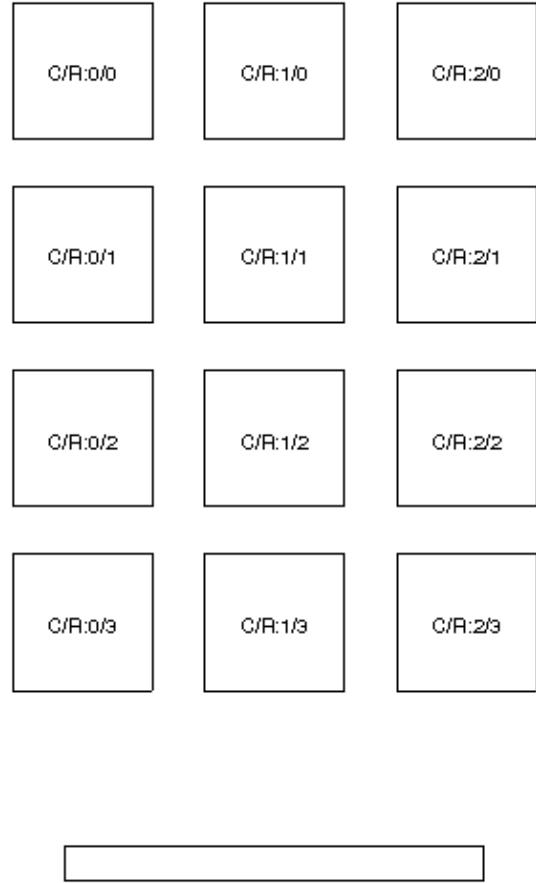
```

```
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.margins.top=.25
M.margins.bottom=.25
M.margins.left=.25
M.margins.right=.25

## The legend uses the bottom margin for display area
## We need to "shrink it"
M.legend.thickness=.1
for i in range(12):
    t=M.get()
    ##     x.plot(s[i],t,iso)
M.preview('test4')

raw_input('Done')
```

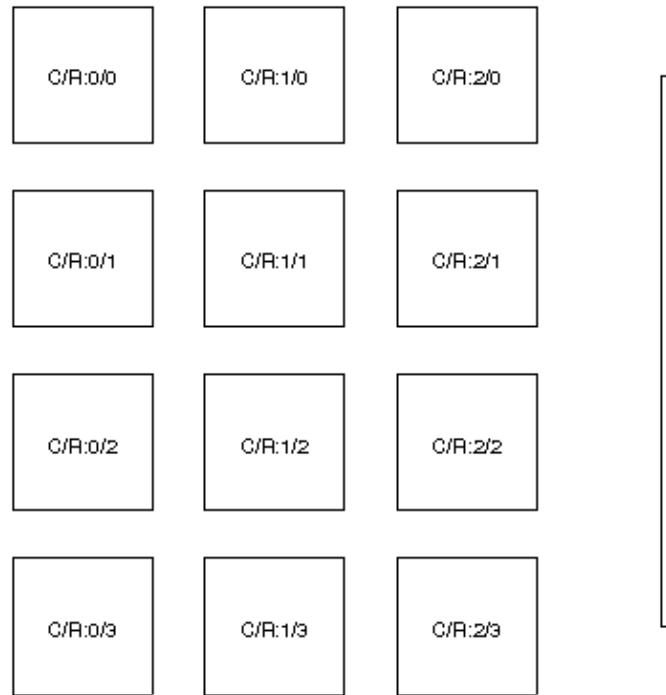


In this example we show how to control margins and legend direction:

```
import EzTemplate,vcs
## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)
M.margins.top=.25
M.margins.bottom=.25
M.margins.left=.25
M.margins.right=.25
```

```
M.legend.direction='vertical'
## The legend uses the right margin for display are
## We need to "shrink it"
M.legend.thickness=.05
for i in range(12):
    t=M.get()
    ##      x.plot(s[i],t,iso)
M.preview('test5')
raw_input('Done')
```



In this example we show how to mix local and global legends, and how to alternate the direction of the local legend bars:

```

import EzTemplate,vcs

## Initialize VCS
x=vcs.init()

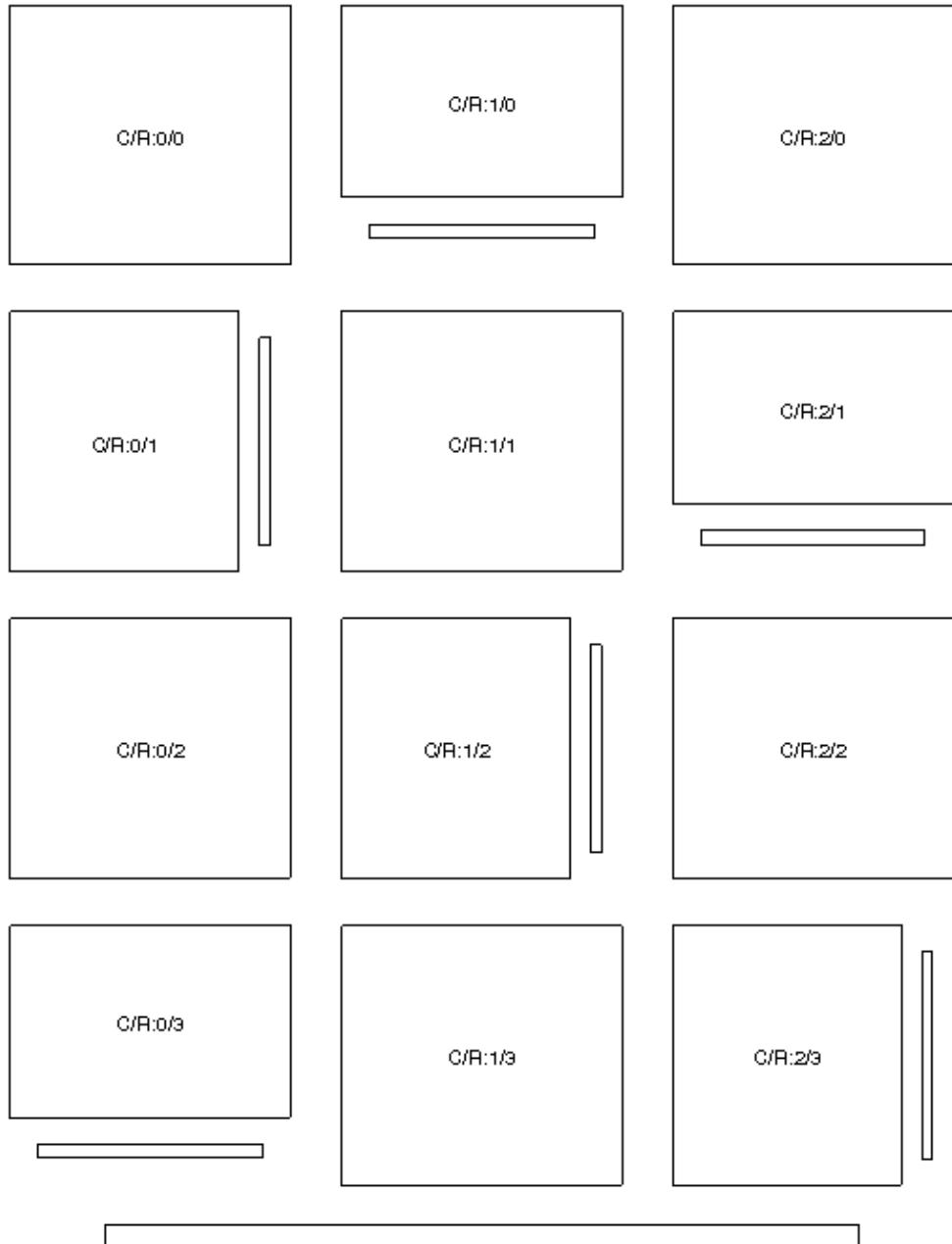
M=EzTemplate.Multi(rows=4,columns=3)
for i in range(12):
    if i%2==1:

```

```

if i%4 == 3:
    M.legend.direction='vertical'
    t=M.get(legend='local')
    M.legend.direction='horizontal'
else:
    t=M.get()
#x.plot(s[i],t,iso)
M.preview('test6')
raw_input('Done')

```



In this example we show how to draw templates in a different order (reversed for this example):

```
import cdms,EzTemplate,vcs,sys

## Initialize VCS
x=vcs.init()

M=EzTemplate.Multi(rows=4,columns=3)

icol=3
irow=4
for i in range(12):
    if i % 3 == 0:
        irow-=1
    icol-=1
    t=M.get(column=icol,row=irow)
#    x.plot(s[11-i],t,iso)
    if icol==0 : icol=3
M.preview('test7')
raw_input('Done')
```



In this example we show how to control the spacing parameter:

```
import EzTemplate,vcs

## Initialize VCS
x=vcs.init()
iso=x.createisofill('test')
iso.levels=vcs.mkscale(0.,100.)
iso.fillareacolors=vcs.getcolors(iso.levels)

M=EzTemplate.Multi(rows=4,columns=3)
```

```
M.spacing.horizontal=.25  
M.spacing.vertical=.1  
M.preview('test8')  
raw_input('Done')
```

